On Coloring Resilient Graphs

Jeremy Kun and Lev Reyzin

Department of Mathematics, Statistics, and Computer Science
University of Illinois at Chicago, Chicago, IL 60607
{jkun2,lreyzin}@math.uic.edu

Abstract. We introduce a new notion of resilience for constraint satisfaction problems, with the goal of more precisely determining the boundary between NP-hardness and the existence of efficient algorithms for resilient instances. In particular, we study r-resiliently k-colorable graphs, which are those k-colorable graphs that remain k-colorable even after the addition of any r new edges. We prove lower bounds on the NP-hardness of coloring resiliently colorable graphs, and provide an algorithm that colors sufficiently resilient graphs. We also analyze the corresponding notion of resilience for k-SAT. This notion of resilience suggests an array of open questions for graph coloring and other combinatorial problems.

1 Introduction and related work

An important goal in studying NP-complete combinatorial problems is to find precise boundaries between tractability and NP-hardness. This is often done by adding constraints to the instances being considered until a polynomial time algorithm is found. For instance, while SAT is NP-hard, the restricted 2-SAT and XOR-SAT versions are decidable in polynomial time.

In this paper we present a new angle for studying the boundary between NP-hardness and tractability. We informally define the resilience of a constraint-based combinatorial problem and we focus on the case of resilient graph colorability. Roughly speaking, a positive instance is resilient if it remains a positive instance up to the addition of a constraint. For example, an instance G of Hamiltonian circuit would be "r-resilient" if G has a Hamiltonian circuit, and G minus any r edges *still* has a Hamiltonian circuit. In the case of coloring, we say a graph G is r-resiliently k-colorable if G is k-colorable and will remain so even if any r edges are added. One would imagine that *finding* a k-coloring in a very resilient graph would be easy, as that instance is very "far" from being not colorable. And in general, one can pose the question: how resilient can instances be and have the search problem still remain hard?[1]

Most NP-hard problems have natural definitions of resiliency. For instance, resilient positive instances for optimization problems over graphs can be defined as those that remain positive instances even up to the addition or removal of

[1] We focus on the search versions of the problems because the decision version on resilient instances induces the trivial "yes" answer.

any edge. For satisfiability, we say a resilient instance is one where variables can be "fixed" and the formula remains satisfiable. In problems like set-cover, we could allow for the removal of a given number of sets. Indeed, this can be seen as a general notion of resilience for adding constraints in constraint satisfaction problems (CSPs), which have an extensive literature [24].[2]

Therefore we focus on a specific combinatorial problem, graph coloring. Resilience is defined up to the addition of edges, and we first show that this is an interesting notion: many famous, well studied graphs exhibit strong resilience properties. Then, perhaps surprisingly, we prove that 3-coloring a 1-resiliently 3-colorable graph is NP-hard – that is, it is hard to color a graph even when it is guaranteed to remain 3-colorable under the addition of any edge. Briefly, our reduction works by mapping positive instances of 3-SAT to 1-resiliently 3-colorable graphs and negative instances to graphs of chromatic number at least 4. An algorithm which can color 1-resiliently 3-colorable graphs can hence distinguish between the two. On the other hand, we observe that 3-resiliently 3-colorable graphs have polynomial-time coloring algorithms (leaving the case of 3-coloring 2-resiliently 3-colorable graphs tantalizingly open). We also show that efficient algorithms exist for k-coloring $\binom{k}{2}$-resiliently k-colorable graphs for all k, and discuss the implications of our lower bounds.

This paper is organized as follows. In the next two subsections we review the literature on other notions of resilience and on graph coloring. In Section 2 we characterize the resilience of boolean satisfiability, which is used in our main theorem on 1-resilient 3-coloring. In Section 3 we formally define the resilient graph coloring problem and present preliminary upper and lower bounds. In Section 4 we prove our main theorem, and in Section 5 we discuss open problems.

1.1 Related work on resilience

There are related concepts of resilience in the literature. Perhaps the closest in spirit is Bilu and Linial's notion of stability [5]. Their notion is restricted to problems over metric spaces; they argue that practical instances often exhibit some degree of stability, which can make the problem easier. Their results on clustering stable instances have seen considerable interest and have been substantially extended and improved [3, 5, 27]. Moreover, one can study TSP and other optimization problems over metrics under the Bilu-Linial assumption [26]. A related notion of stability by Ackerman and Ben-David [1] for clustering yields efficient algorithms when the data lies in Euclidian space.

Our notion of resilience, on the other hand, is most natural in the case when the optimization problem has natural constraints, which can be fixed or modified. Our primary goal is also different – we seek to more finely delineate the boundary between tractability and hardness in a systematic way across problems.

Property testing can also be viewed as involving resilience. Roughly speaking property testing algorithms distinguish between combinatorial structures that

[2] However, a resilience definition for general CSPs is not immediate because the ability to add any constraint (e.g., the negation of an existing constraint) is too strong.

satisfy a property or are very far from satisfying it. These algorithms are typically given access to a small sample depending on a parameter ε alone. For graph property testing, as with resilience, the concept of being ε-far from having a property involves the addition or removal of an arbitrary set of at most $\varepsilon\binom{n}{2}$ edges from G. Our notion of resilience is different in that we consider adding or removing a constant number of constraints. More importantly, property testing is more concerned with query complexity than with computational hardness.

1.2 Previous work on coloring

As our main results are on graph colorability, we review the relevant past work. A graph G is k-colorable if there is an assignment of k distinct colors to the vertices of G so that no edge is monochromatic. Determining whether G is k-colorable is a classic an NP-hard problem [19]. Many attempts to simplify the problem, such as assuming planarity or bounded degree, still result in NP-hardness [8]. A large body of work surrounds positive and negative results for explicit families of graphs. The list of families that are polynomial-time colorable includes triangle-free planar graphs, perfect graphs and almost-perfect graphs, bounded tree- and clique-width graphs, quadtrees, and various families of graphs defined by the lack of an induced subgraph [7, 10, 15, 22, 23].

With little progress on coloring general graphs, research has naturally turned to approximation. In approximating the chromatic number of a general graph, the first results were of Garey and Johnson, giving a performance guarantee of $O(n/\log n)$ colors [18] and proving that it is NP-hard to approximate chromatic number to within a constant factor less than two [11]. Further work improved this bound by logarithmic factors [4, 13]. In terms of lower bounds, Zuckerman [29] derandomized the PCP-based results of Håstad [14] to prove the best known approximability lower-bound to date, $O(n^{1-\varepsilon})$.

There has been much recent interest in coloring graphs which are already known to be colorable while minimizing the number of colors used. For a 3-colorable graph, Wigderson gave an algorithm using at most $O(n^{1/2})$ colors [28], which Blum improved to $\tilde{O}(n^{3/8})$ [6]. A line of research improved this bound still further to $o(n^{1/5})$ [17]. Despite the difficulties in improving the constant in the exponent, and as suggested by Arora [2], there is no evidence that coloring a 3-colorable graph with as few as $O(\log n)$ colors is hard.

On the other hand there are asymptotic and concrete lower bounds. Khot [21] proved that for sufficiently large k it is NP-hard to color a k-colorable graph with fewer than $k^{O(\log k)}$ colors; this was improved by Huang to $2^{\sqrt[3]{k}}$ [16]. It is also known that for every constant h there exists a sufficiently large k such that coloring a k-colorable graph with hk colors is NP-hard [9]. In the non-asymptotic case, Khanna, Linial, and Safra [20] used the PCP theorem to prove it is NP-hard to 4-color a 3-colorable graph, and more generally to color a k colorable graph with at most $k + 2\lfloor k/3 \rfloor - 1$ colors. Guruswami and Khanna give an explicit reduction for $k = 3$ [12]. Assuming a variant of Khot's 2-to-1 conjecture, Dinur et al. prove that distinguishing between chromatic number K and K' is hard for constants $3 \leq K < K'$ [9]. This is the best conditional lower bound we give in Section 3.3, but it does not to our knowledge imply Theorem 2.

Without large strides in approximate graph coloring, we need a new avenue to approach the NP-hardness boundary. In this paper we consider the coloring problem for a general family of graphs which we call *resiliently colorable*, in the sense that adding edges does not violate the given colorability assumption.

2 Resilient SAT

We begin by describing a resilient version of k-satisfiability, which is used in proving our main result for resilient coloring in Section 4.

Problem 1 (resilient k-SAT) *A boolean formula φ is* r-resilient *if it is satisfiable and remains satisfiable if any set of r variables are fixed. We call* r-resilient k-SAT *the problem of finding a satisfying assignment for an r-resiliently satisfiable k-CNF formula. Likewise, r-resilient CNF-SAT is for r-resilient formulas in general CNF form.*

Note that this definition is equivalent to one where you may add arbitrary clauses of size k, because fixing a variable is always a stronger constraint than requiring the truth of a disjunction of more than one variable.

The following lemma allows us to take problems that involve low (even zero) resilience and blow them up to have large resilience and large clause size.

Lemma 1 (blowing up). *For all $r \geq 0$, $s \geq 1$, and $k \geq 3$, r-resilient k-SAT reduces to $[(r+1)s-1]$-resilient (sk)-SAT in polynomial time.*

Proof. Let φ be an r-resilient k-SAT formula. For each i, let φ^i denote a copy of φ with a fresh set of variables. Construct $\psi = \bigvee_{i=1}^{s} \varphi^i$. The formula ψ is clearly equivalent to φ, and by distributing the terms we can transform ψ into (sk)-CNF form in time $O(n^s)$. We claim that ψ is $[(r+1)s-1]$-resilient. If fewer than $(r+1)s$ variables are fixed, then by the pigeonhole principle one of the s sets of variables has at most r fixed variables. Suppose this is the set for φ^1. As φ is r-resilient, φ^1 is satisfiable and hence so is ψ. $\square$

As a consequence of the blowing up lemma for $r = 0, s = 2, k = 3$, 1-resilient 6-SAT is NP-hard (we reduce from this in our main coloring lower bound). Moreover, a slight modification of the proof shows that r-resilient CNF-SAT is NP-hard for all $r \geq 0$. The next lemma allows us to reduce in the other direction, shrinking down the resilience and clause sizes.

Lemma 2 (shrinking down). *Let $r \geq 1$, $k \geq 2$, and $q = \min(r, \lfloor k/2 \rfloor)$. Then r-resilient k-SAT reduces to q-resilient $(\lceil \frac{k}{2} \rceil + 1)$-SAT in polynomial time.*

Proof. For ease of notation, we prove the case where k is even. For a clause $C = \bigvee_{i=1}^{k} x_i$, denote by $C[:k/2]$ the sub-clause consisting of the first half of the literals of C, specifically $\bigvee_{i=1}^{k/2} x_i$. Similarly denote by $C[k/2:]$ the second half of C. Now given a k-SAT formula $\varphi = \bigwedge_{j=1}^m C_j$, we construct a $(\frac{k}{2}+1)$-SAT formula ψ by the following. For each j introduce a new variable z_j, and define

$$\psi = \bigwedge_{j=1}^{m} (C_j[: k/2] \vee z_j) \wedge (C_j[k/2 :] \vee \overline{z_j})$$

The formulas φ and ψ are logically equivalent, and we claim ψ is q-resilient. Indeed, if some of the original set of variables are fixed there is no problem, and each z_i which is fixed corresponds to a choice of whether the literal which will satisfy C_j comes from the first or the second half. Even stronger, we can arbitrarily *pick* another literal in the correct half and fix its variable so as to satisfy the clause. The r-resilience of φ guarantees the ability to do this for up to r of the z_i. But with the observation that there are no l-resilient l-SAT formulas, we cannot get $k/2+1$ resilience when $r > k/2$, giving the definition of q. $\square$

Combining the blowing up and shrinking down lemmas, we get a tidy characterization: r-resilient k-SAT is either NP-hard or vacuously trivial.

Theorem 1 *For all $k \geq 3$, $0 \leq r < k$, r-resilient k-SAT is NP-hard.*

Proof. We note that increasing k or decreasing r (while leaving the other parameter fixed) cannot make r-resilient k-SAT easier, so it suffices to reduce from 3-SAT to $(k-1)$-resilient k-SAT for all $k \geq 3$. For any r we can blow up from 3-SAT to r-resilient $3(r+1)$-SAT by setting $s = r+1$ in the blowing up lemma. We want to iteratively apply the shrinking down lemma until the clause size is s. If we write $s_0 = 3s$ and $s_i = \lceil s_i/2 \rceil + 1$, we would need that for some m, $s_m = s$ and that for each $1 \leq j < m$, the inequality $\lfloor s_j/2 \rfloor \geq r = s-1$ holds.

Unfortunately this is not always true. For example, if $s = 10$ then $s_1 = 16$ and $16/2 < 9$, so we cannot continue. However, we can avoid this for sufficiently large r by artificially increasing k after blowing up. Indeed, we just need to find some $x \geq 0$ for which $a_1 = \left\lceil \frac{3s+x}{2} \right\rceil + 1 = 2(s-1)$. And we can pick $x = s-6 = r-5$, which works for all $r \geq 5$. For $r = 2, 3, 4$, we can check by hand that one can find an x that works.[3] For $r = 2$ we can start from 2-resilient 9-SAT; for $r = 3$ we can start from 16-SAT; and for $r = 4$ we can start from 24-SAT. $\square$

3 Resilient graph coloring and preliminary bounds

In contrast to satisfiability, resilient graph coloring has a more interesting hardness boundary, and it is not uncommon for graphs to have relatively high resilience. In this section we present some preliminary bounds.

[3] The difference is that for $r \geq 5$ we can get what we need with only two iterations, but for smaller r we require three steps.

3.1 Problem definition and remarks

Problem 2 (resilient coloring) *A graph G is called* r-resiliently k-colorable *if G remains k-colorable under the addition of any set of r new edges.*

We argue that this notion is not trivial by showing the resilience properties of some classic graphs. These were determined by exhaustive computer search. The Petersen graph is 2-resiliently 3-colorable. The Dürer graph is 1-resiliently 3-colorable (but not 2-resilient) and 4-resiliently 4-colorable (but not 5-resilient). The Grötzsch graph is 4-resiliently 4-colorable (but not 5-resilient). The Chvátal graph is 3-resiliently 4-colorable (but not 4-resilient).

There are a few interesting constructions to build intuition about resilient graphs. First, it is clear that every k-colorable graph is 1-resiliently $(k+1)$-colorable (just add one new color for the additional edge), but for all $k > 2$ there exist k-colorable graphs which are not 2-resiliently $(k+1)$-colorable. Simply remove two disjoint edges from the complete graph on $k+2$ vertices. A slight generalization of this argument provides examples of graphs which are $\lfloor (k+1)/2 \rfloor$-colorable but not $\lfloor (k+1)/2 \rfloor$-resiliently k-colorable for $k \geq 3$. On the other hand, every $\lfloor (k+1)/2 \rfloor$-colorable graph is $(\lfloor (k+1)/2 \rfloor - 1)$-resiliently k-colorable, since r-resiliently k-colorable graphs are $(r+m)$-resiliently $(k+m)$-colorable for all $m \geq 0$ (add one new color for each added edge).

One expects high resilience in a k-colorable graph to reduce the number of colors required to color it. While this may be true for super-linear resilience, there are easy examples of $(k-1)$-resiliently k-colorable graphs which are k-chromatic. For instance, add an isolated vertex to the complete graph on k vertices.

3.2 Observations

We are primarily interested in the complexity of coloring resilient graphs, and so we pose the question: for which values of k, r does the task of k-coloring an r-resiliently k-colorable graph admit an efficient algorithm? The following observations aid us in the classification of such pairs, which is displayed in Figure 1.

Observation 1 *An r-resiliently k-colorable graph is r'-resiliently k-colorable for any $r' \leq r$. Hence, if k-coloring is in P for r-resiliently k-colorable graphs, then it is for s-resiliently k-colorable graphs for all $s \geq r$. Conversely, if k-coloring is NP-hard for r-resiliently k-colorable graphs, then it is for s-resiliently k-colorable graphs for all $s \leq r$.*

Hence, in Figure 1 if a cell is in P, so are all of the cells to its right; and if a cell is NP-hard, so are all of the cells to its left.

Observation 2 *If k-coloring is in P for r-resiliently k-colorable graphs, then k'-coloring r-resiliently k'-colorable graphs is in P for all $k' \leq k$. Similarly, if k-coloring is in NP-hard for r-resiliently k-colorable graphs, then k'-coloring is NP-hard for r-resiliently k'-colorable graphs for all $k' \geq k$.*

Proof. If G is r-resiliently k-colorable, then we construct G' by adding a new vertex v with complete incidence to G. Then G' is r-resiliently $(k+1)$-colorable, and an algorithm to color G' can be used to color G. □

Observation 2 yields the rule that if a cell is in P, so are all of the cells above it; if a cell is NP-hard, so are the cells below it. More generally, we have the following observation which allows us to apply known bounds.

Observation 3 *If it is NP-hard to $f(k)$-color a k-colorable graph, then it is NP-hard to $f(k)$-color an $(f(k)-k)$-resiliently $f(k)$-colorable graph.*

This observation is used in Propositions 2 and 3, and follows from the fact that an r-resiliently k-colorable graph is $(r+m)$-resiliently $(k+m)$-colorable for all $m \geq 0$ (here $r = 0, m = f(k) - k$).

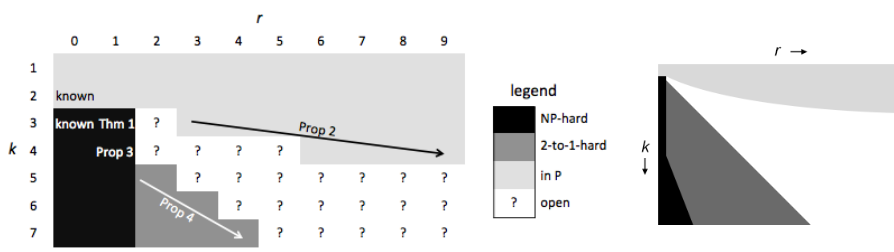


Fig. 1: The classification of the complexity of k-coloring r-resiliently k-colorable graphs. Left: the explicit classification for small k, r. Right: a zoomed-out view of the same table, with the NP-hard (black) region added by Proposition 4.

3.3 Upper and lower bounds

In this section we provide a simple upper bound on the complexity of coloring resilient graphs, we apply known results to show that 4-coloring a 1-resiliently 4-colorable graph is NP-hard, and we give the conditional hardness of k-coloring $(k-3)$-resiliently k-colorable graphs for all $k \geq 3$. This last result follows from the work of Dinur et al., and depends a variant of Khot's 2-to-1 conjecture [9]; a problem is called *2-to-1-hard* if it is NP-hard assuming this conjecture holds. Finally, applying the result of Huang [16], we give an asymptotic lower bound.

All our results on coloring are displayed in Figure 1. To explain Figure 1 more explicitly, Proposition 1 gives an upper bound for $r = \binom{k}{2}$, and Proposition 2 gives hardness of the cell $(4, 1)$ and its consequences. Proposition 3 provides the conditional lower bound, and Theorem 2 gives the hardness of the cell $(3, 1)$. Proposition 4 provides an NP-hardness result.

Proposition 1 *There is an efficient algorithm for k-coloring $\binom{k}{2}$-resiliently k-colorable graphs.*

Proof. If G is $\binom{k}{2}$-resiliently k-colorable, then no vertex may have degree $\geq k$. For if v is such a vertex, one may add complete incidence to any choice of k vertices in the neighborhood of v to get K_{k+1}. Finally, graphs with bounded degree $k-1$ are greedily k-colorable. $\square$

Proposition 2 *4-coloring a 1-resiliently 4-colorable graph is NP-hard.*

Proof. It is known that 4-coloring a 3-colorable graph is NP-hard, so we may apply Observation 3. Every 3-colorable graph G is 1-resiliently 4-colorable, since if we are given a proper 3-coloring of G we may use the fourth color to properly color any new edge that is added. So an algorithm A which efficiently 4-colors 1-resiliently 4-colorable graphs can be used to 4-color a 3-colorable graph. $\square$

Proposition 3 *For all $k \geq 3$, it is 2-to-1-hard to k-color a $(k-3)$-resiliently k-colorable graph.*

Proof. As with Proposition 2, we apply Observation 3 to the conditional fact that it is NP-hard to k-color a 3-colorable graph for $k > 3$. Such graphs are $(k-3)$-resiliently k-colorable. $\square$

Proposition 4 *For sufficiently large k it is NP-hard to $2^{\sqrt[3]{k}}$-color an r-resiliently $2^{\sqrt[3]{k}}$-colorable graph for $r < 2^{\sqrt[3]{k}} - k$.*

Proposition 4 comes from applying Observation 3 to the lower bound of Huang [16]. The only unexplained cell of Figure 1 is (3,1), which we prove is NP-hard as our main theorem in the next section.

4 NP-hardness of 1-resilient 3-colorability

Theorem 2 *It is NP-hard to 3-color a 1-resiliently 3-colorable graph.*

Proof. We reduce 1-resilient 3-coloring from 1-resilient 6-SAT. This reduction comes in the form of a graph which is 3-colorable if and only if the 6-SAT instance is satisfiable, and 1-resiliently 3-colorable when the 6-SAT instance is 1-resiliently satisfiable. We use the colors white, black, and gray.

We first describe the gadgets involved and prove their consistency (that the 6-SAT instance is satisfiable if and only if the graph is 3-colorable), and then prove the construction is 1-resilient. Given a 6-CNF formula $\varphi = C_1 \wedge \cdots \wedge C_m$ we construct a graph G as follows. Start with a base vertex b which we may assume w.l.o.g. is always colored gray. For each literal we construct a *literal gadget* consisting of two vertices both adjacent to b, as in Figure 2. As such, the vertices in a literal gadget may only assume the colors white and black. A variable is interpreted as true iff both vertices in the literal gadget have the same color. We will abbreviate this by saying a literal is *colored true* or *colored false*.

We connect two literal gadgets for $x, \overline{x}$ by a *negation gadget* in such a way that the gadget for x is colored true if and only if the gadget for $\overline{x}$ is colored false.

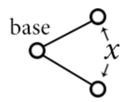


Fig. 2: The gadget for a literal. The two single-degree vertices represent a single literal, and are interpreted as true if they have the same color. The base vertex is always colored gray. Note this gadget comes from Kun et al. [25].

The negation gadget is given in Figure 3. In the diagram, the vertices labeled 1 and 3 correspond to x, and those labeled 10 and 12 correspond to $\overline{x}$. We start by showing that no proper coloring can exist if both literal gadgets are colored true. If all four of these vertices are colored white or all four are black, then vertices 6 and 7 must also have this color, and so the coloring is not proper. If one pair is colored both white and the other both black, then vertices 13 and 14 must be gray, and the coloring is again not proper. Next, we show that no proper coloring can exist if both literal gadgets are colored false. First, if vertices 1 and 10 are white and vertices 3 and 12 are black, then vertices 2 and 11 must be gray and the coloring is not proper. If instead vertices 1 and 12 are white and vertices 3 and 10 black, then again vertices 13 and 14 must be gray. This covers all possibilities up to symmetry. Moreover, whenever one literal is colored true and the other false, one can extend it to a proper 3-coloring of the whole gadget.

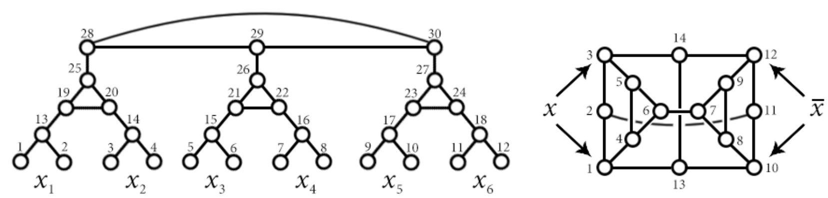
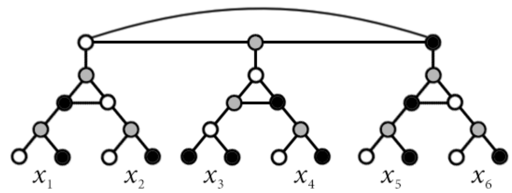


Fig. 3: Left: the gadget for a clause. Right: the negation gadget ensuring two literals assume opposite truth values.

Now suppose we have a clause involving literals, w.l.o.g., $x_1, \ldots, x_6$. We construct the *clause gadget* shown in Figure 3, and claim that this gadget is 3-colorable iff at least one literal is colored true. Indeed, if the literals are all colored false, then the vertices 13 through 18 in the diagram must be colored gray, and then the vertices 25, 26, 27 must be gray. This causes the central triangle to use only white and black, and so it cannot be a proper coloring. On the other hand, if some literal is colored true, we claim we can extend to a proper coloring of the whole gadget. Suppose w.l.o.g. that the literal in question is x_1, and that vertices 1 and 2 both are black. Then Figure 4 shows how this extends to a proper coloring of the entire gadget regardless of the truth assignments of the other literals (we can always color their branches as if the literals were false).

It remains to show that G is 1-resiliently 3-colorable when φ is 1-resiliently satisfiable. This is because a new edge can, at worst, fix the truth assignment (perhaps indirectly) of at most one literal. Since the original formula φ is 1-

Fig. 4: A valid coloring of the clause gadget when one variable (in this case x_3) is true.



resiliently satisfiable, G maintains 3-colorability. Additionally, the gadgets and the representation of truth were chosen so as to provide flexibility w.r.t. the chosen colors for each vertex, so many edges will have no effect on G's colorability.

First, one can verify that the gadgets themselves are 1-resiliently 3-colorable.[4] We break down the analysis into eight cases based on the endpoints of the added edge: within a single clause/negation/literal gadget, between two distinct clause/negation/literal gadgets, between clause and negation gadgets, and between negation and literal gadgets. We denote the added edge by $e = (v, w)$ and call it *good* if G is still 3-colorable after adding e.

Literal Gadgets. First, we argue that e is good if it lies within or across literal gadgets. Indeed, there is only one way to add an edge within a literal gadget, and this has the effect of setting the literal to false. If e lies across two gadgets then it has no effect: if c is a proper coloring of G without e, then after adding e either c is still a proper coloring or we can switch to a different representation of the truth value of v or w to make e properly colored (i.e. swap "white white" with "black black," or "white black" with "black white" and recolor appropriately).

Negation Gadgets. Next we argue that e is good if it involves a negation gadget. Let N be a negation gadget for the variable x. Indeed, by 1-resilience an edge within N is good; e only has a local effect within negation gadgets, and it may result in fixing the truth value of x. Now suppose e has only one vertex v in N. Figure 5 shows two ways to color N, which together with reflections along the horizontal axis of symmetry have the property that we may choose from at least two colors for any vertex we wish. That is, if we are willing to fix the truth value of x, then we may choose between one of two colors for v so that e is properly colored regardless of which color is adjacent to it.

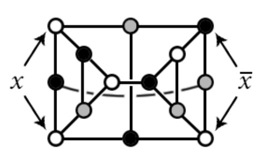


Fig. 5: Two distinct ways to color a negation gadget without changing the truth values of the literals. Only the rightmost center vertex cannot be given a different color by a suitable switch between the two representations or a reflection of the graph across the horizontal axis of symmetry. If the new edge involves this vertex, we must fix the truth value appropriately.

[4] These graphs are small enough to admit verification by computer search.

Clause Gadgets. Suppose e lies within a clause gadget or between two clause gadgets. As with the negation gadget, it suffices to fix the truth value of one variable suitably so that one may choose either of two colors for one end of the new edge. Figure 6 provides a detailed illustration of one case. Here, we focus on two branches of two separate clause gadgets, and add the new edge $e = (v, w)$. The added edge has the following effect: if x is false, then neither y nor z may be used to satisfy C_2 (as w cannot be gray). This is no stronger than requiring that either x be true or y and z both be false, i.e., we add the clause $x \vee (\overline{y} \wedge \overline{z})$ to φ. This clause can be satisfied by fixing a single variable (x to true), and φ is 1-resilient, so we can still satisfy φ and 3-color G. The other cases are analogous.

This proves that G is 1-resilient when φ is, and finishes the proof. □

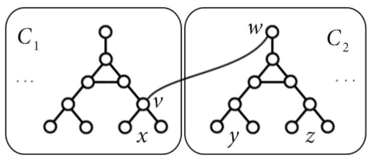


Fig. 6: An example of an edge added between two clauses C_1, C_2.

5 Discussion and open problems

The notion of resilience introduced in this paper leaves many questions unanswered, both specific problems about graph coloring and more general exploration of resilience in other combinatorial problems and CSPs.

Regarding graph coloring, our paper established the fact that 1-resilience doesn't affect the difficulty of graph coloring. However, the question of 2-resilience is open, as is establishing linear lower bounds without dependence on the 2-to-1 conjecture. There is also room for improvement in finding efficient algorithms for highly-resilient instances, closing the gap between NP-hardness and tractability.

On the general side, our framework applies to many NP-complete problems, including Hamiltonian circuit, set cover, 3D-matching, integer LP, and many others. Each presents its own boundary between NP-hardness and tractability, and there are undoubtedly interesting relationships across problems.

Acknowledgments. We thank Shai Ben-David for helpful discussions.

References

1. Margareta Ackerman and Shai Ben-David. Clusterability: A theoretical study. *Journal of Machine Learning Research - Proceedings Track*, 5:1–8, 2009.
2. Sanjeev Arora and Rong Ge. New tools for graph coloring. In *APPROX-RANDOM*, pages 1–12, 2011.
3. Pranjal Awasthi, Avrim Blum, and Or Sheffet. Center-based clustering under perturbation stability. *Inf. Process. Lett.*, 112(1-2):49–54, 2012.
4. Bonnie Berger and John Rompel. A better performance guarantee for approximate graph coloring. *Algorithmica*, 5(3):459–466, 1990.

5. Yonatan Bilu and Nathan Linial. Are stable instances easy? *Combinatorics, Probability & Computing*, 21(5):643–660, 2012.
6. Avrim Blum. New approximation algorithms for graph coloring. *J. ACM*, 41(3):470–516, 1994.
7. Leizhen Cai. Parameterized complexity of vertex colouring. *Discrete Applied Mathematics*, 127(3):415–429, 2003.
8. David P. Dailey. Uniqueness of colorability and colorability of planar 4-regular graphs are np-complete. *Discrete Mathematics*, 30(3):289 – 293, 1980.
9. Irit Dinur, Elchanan Mossel, and Oded Regev. Conditional hardness for approximate coloring. *SIAM J. Comput.*, 39(3):843–873, 2009.
10. David Eppstein, Marshall W. Bern, and Brad L. Hutchings. Algorithms for coloring quadtrees. *Algorithmica*, 32(1):87–94, 2002.
11. Michael R. Garey and David S. Johnson. The complexity of near-optimal graph coloring. *J. ACM*, 23(1):43–49, 1976.
12. Venkatesan Guruswami and Sanjeev Khanna. On the hardness of 4-coloring a 3-colorable graph. *SIAM J. Discrete Math.*, 18(1):30–40, 2004.
13. Magnús M. Halldórsson. A still better performance guarantee for approximate graph coloring. *Inf. Process. Lett.*, 45(1):19–23, 1993.
14. Johan Håstad. Clique is hard to approximate within $n^{1-\varepsilon}$. *Acta Mathematica*, 182:105–142, 1999.
15. Chính T. Hoàng, Frédéric Maffray, and Meriem Mechebbek. A characterization of b-perfect graphs. *Journal of Graph Theory*, 71(1):95–122, 2012.
16. Sangxia Huang. Improved hardness of approximating chromatic number. *CoRR*, abs/1301.5216, 2013.
17. Ken ichi Kawarabayashi and Mikkel Thorup. Coloring 3-colorable graphs with $o(n^{1/5})$ colors. In *STACS*, volume 25, pages 458–469, 2014.
18. David S. Johnson. Worst case behavior of graph coloring algorithms. In *Proc. 5th Southeastern Conf. on Comb., Graph Theory and Comput.*, pages 513–527, 1974.
19. Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972.
20. Sanjeev Khanna, Nathan Linial, and Shmuel Safra. On the hardness of approximating the chromatic number. *Combinatorica*, 20(3):393–415, 2000.
21. Subhash Khot. Improved inaproximability results for maxclique, chromatic number and approximate graph coloring. In *FOCS*, pages 600–609, 2001.
22. Daniel Kobler and Udi Rotics. Edge dominating set and colorings on graphs with fixed clique-width. *Discrete Applied Mathematics*, 126(2-3):197–221, 2003.
23. Daniel Král, Jan Kratochvíl, Zsolt Tuza, and Gerhard J. Woeginger. Complexity of coloring graphs without forbidden induced subgraphs. In *WG*, pages 254–262, 2001.
24. Vipin Kumar. Algorithms for constraint-satisfaction problems: A survey. *AI magazine*, 13(1):32, 1992.
25. Jeremy Kun, Brian Powers, and Lev Reyzin. Anti-coordination games and stable graph colorings. In *SAGT*, pages 122–133, 2013.
26. Matús Mihalák, Marcel Schöngens, Rastislav Srámek, and Peter Widmayer. On the complexity of the metric tsp under stability considerations. In *SOFSEM*, pages 382–393, 2011.
27. Lev Reyzin. Data stability in clustering: A closer look. In *ALT*, pages 184–198, 2012.
28. Avi Wigderson. Improving the performance guarantee for approximate graph coloring. *J. ACM*, 30(4):729–735, 1983.

29. David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(1):103–128, 2007.